

A Legacy Controlled Materials Tracking System at LLNL:

Maintenance and Testing

Charles Parrish

Wally Yee

Gayle Zeigler

1.0 Introduction

Controlled Materials Tracking System (COMATS) is responsible for tracking controlled material at the Lawrence Livermore National Laboratory. Types of controlled material tracked include nuclear materials, precious metals, parts classified by shape, sealed radioactive sources, and stable isotopes. The inventories of this material must be tracked for a variety of health, safety, environmental and national security reasons.

COMATS is a legacy system dating back to 1988. It was first deployed on a network of DEC MicroVAX IIs under the VMS operating system and Ingres Data Base Management System (DBMS). It currently resides on a DEC VAX Cluster of two 4000/300. All of the applications that make up COMATS use the Ingres Data Base Management System. Most of the software was written in C, but there are also some FORTRAN and COBOL modules that are used to produce reports.

COMATS is a highly interactive system. All interaction is menu and forms based. Updates are performed in the form of transactions, which from the users point-of-view, represent a physical operation, e.g., movement, split, etc., of material.

This paper deals with the maintenance and testing of COMATS and is organized as follows: Section 1) Introduction. Section 2) Maintenance of COMATS. Section 3) Testing process. Section 4) Release Documentation. Section 5) Installation of new software on the production system.

2.0 Maintenance: Changes and Releases

All changes to COMATS are controlled by a Configuration Management Plan and are done under the auspices of a Change Request. There are three types of Change Requests: Software Change Requests (SCRs) are used to authorize changes to the production software and some critical reports; Report Change Requests (RCRs) are used to authorize new reports or changes to existing reports; Data Change Requests (DCRs) are used to authorize changes to the data base itself, including configuration and authorization data.

Implementation of Software Change Requests are those that have the greatest impact on the data base because they authorize the modification of software that controls the data going into the data base. For that reason, formal testing of all software changes is required. Since testing is such a significant effort, the implementation of several SCRs are grouped together into what is referred to as a release. All development and testing of the implementation of SCRs is done on MMTEST, the test data base. After independent testing and validation have assured that the new release is behaving properly, the upgraded software is installed on the production system.

After making the decision as to which SCRs are intended for the release, a typical release cycle consists of the following steps: The developer 1) Writes implementation plans for each of the SCRs, indicating which software modules, forms, reports, tables, and command files are affected by each SCR. 2) Provides a description of the unit test cases for each SCR which will be incorporated into the formal test suite. 3) Reserves appropriate modules from the software library. 4) Modifies the existing modules or writes new ones, as necessary to implement the SCRs. 5) Tests the implementation to assure that it is correct. 6) Turns the completed modules, along with the implementation plans, to a second developer for walk-through or code review. 7) Replaces

Maintenance/Testing of a Legacy Controlled Materials Tracking System at LLNL

existing modules and adds new modules to the software library. After all SCRs have been implemented: 1) The system manager "builds" the application system, compiling all modified and new modules from the library and linking them. 2) Independent testers execute the test suites described in the next section. 3) If the release passes the test suites, the system manager places it into production.

3.0 Testing Process

For each release, the testing effort is in parallel with the development effort. Each release requires its own Test Plan and Test Procedure, which are written using the IEEE 829-1983 Standard. The Test Plan describes the planned testing process which provides: a summary of the baselines to be produced; a data flow diagram of the process from the first regression baseline through the final SCR-specific baseline; and an estimated schedule.

The test procedure consists of two types of test suites: 1) a regression test suite, which tests all pertinent features of the existing software and is used to assure that the functionality, performance and security of the existing software will not be adversely affected by the implementation of software changes; 2) a SCR-specific test suite, which tests all new software features and assures that the implementation of the SCRs specified above is correct.

3.1 Baselines

A baseline consists of the hardware and software configuration to be tested, the initial state data base, a test suite to be run, and the final state of the data base. Each release may require several baselines to be produced. For a release implementing a change to only the application software, a minimum of three baselines are required: 1) A regression baseline which runs the regression test suite against the current production software; 2) a baseline which runs the same test suite against the software to be released; and 3) a baseline which runs a suite specifically designed to test the implementation of the SCRs in the release being tested.

Releases that include hardware and/or operating system and/or data base management system upgrades may require additional baselines running the regression test suite. This is done to localize problems that may occur with the upgrades.

3.2 Test Suites

Each test suite is divided into one or more test cases, with each test case designed to test a single pertinent feature of the software. Each test case is divided into one or more sessions, where each session consists of a login, a series of transactions to be performed, and a logoff.

Test cases are designed to be independent, and can be run in any order, with a few exceptions; these exceptions are needed to get initial or final reports, and to place the data base in a desired initial or final state. Within a test case, however, the sessions and transactions within a session must be run in the specified order. Different test cases may be run in parallel, and a tester may be asked to switch from one test case to another.

The configuration of MMTEST is very similar to that of MMA, the production data base. MMTEST contains several "user" accounts that mimic the behavior of the real user accounts on MMA. For each session in a test case, the tester logs into MMTEST as the specified user and performs the transactions that are specified.

Each test case consists of two types of Excel documents. The first, called a "workbook" roughly corresponds to the test design document in the IEEE 829-1983 standard. It is a list of sessions and, for each session, a list of transactions to be performed. The workbook defines the data to be used, contains the mathematical formulas necessary to predict what the tester will see on the screen when performing a transaction, which models the behavior of the system. For example, if the

Maintenance/Testing of a Legacy Controlled Materials Tracking System at LLNL

tester is asked to split an item, and in the course of the transaction, COMATS displays default values for the products, the workbook contains the calculations necessary to predict these defaults.

The second type of document is the test script, which is a step-by-step procedure which the tester follows; there is one test script for each session. Since the tester need not be familiar with COMATS, VAX/VMS, or Ingres, each step in the script specifies which value to type or function key to press, and predicts the results of that entry.

The workbook and the test scripts are linked; the test script contains references to cells in the workbook. In this way, changes can be made to the workbook and all associated references in the test scripts will be automatically updated.

3.3 Validation

Validation is the process which assures that the software under test conditions has behaved properly. It compares the state of the data base following the execution of a test suite with an expected state. Regression validation compares the state of the data base for the second baseline (new software) with the state of the data base for the first baseline (current software). For SCR-specific validation, there is nothing to compare the final state of the data base with; therefore, manual inspection is necessary. In addition, the nuclear materials data base has inherent redundancies which must be checked to assure that consistency of the data base has been maintained.

4.0 Release Documentation

Several documents are produced during a release. These include the test plan, workbook, and test scripts described above. In addition, Test Incident Reports are generated throughout the testing process.

Test Incident Reports (TIRs) are used to log abnormal incidents that occur during the testing and validation process. A TIR describes a problem that occurred, has space for comments on the part of the developer, and has space for describing the resolution. Many TIRs are the result of tester error, and their resolution so indicates.

Some TIRs are the result of faults found in the software during the testing process. These TIRs are resolved by either correcting the fault and retesting, or writing a Software Change Request to correct the fault in a future release. All TIRs must be resolved before final release by either correcting the fault, generating an SCR or explaining why neither of these options is necessary.

The final test document is a validation report which contains a statement that the software release has passed all tests and may be placed into production. This statement is supported by: 1) describing the testing and validation process; 2) summarizing problems that occurred during the testing and validation process; and 3) explaining all anomalies and their resolution.

5.0 Cutover

The cutover process transfers all released software from the test system to the production system and makes changes to the data base, if necessary. The released software includes executables, command procedures, forms, and reports. Data base changes include adding new tables, views, and makes structure changes to the existing tables and views. A complete backup of the production system to tape and a disk backup copy of the current software and data base are created before the cutover.

A set of command procedures is used to transfer the software to the production system and setup the software access protection. If any of the data base tables' structure is changed in the release, the data base table contents are downloaded to a text file before and after the change to verify that the structure change on the table does not alter the data. After the cutover process is completed, the new production system is backed up to tape.

Maintenance/Testing of a Legacy Controlled Materials Tracking System at LLNL

This work was performed under the auspices of the U.S. Dept of Energy at LLNL under contract no. W-7405-Eng-48.

Charles Parrish - (510) 422-1306 - parrish3@llnl.gov

Charles Parrish has a Masters Degree in Computer Science from the University of California at Davis. He has been working at the Lawrence Livermore National Laboratory for 20 years, and has been in his current project for 6 years. Charles has written several of the Test Plans, Test Procedures, and Validation Reports mentioned in this paper.

Wally Yee - (510) 422-8398 - yee3@llnl.gov

Wally Yee has a Bachelors Degree in Computer Science and Physics from Oregon State University. He has worked in Hanford for 10 years and at Lawrence Livermore National Laboratory for the past 6 years. He provides software maintenance, user support, system support, and data base administration support for the COMATS system.

Gayle Zeigler - (510) 423-5931 - zeigler2@llnl.gov

Gayle Zeigler has a Bachelors Degree in Computer Information System from the University of California Hayward. She has worked in the Quality Assurance field since 1976, beginning with Quality Assurance/Control in manufacturing and expanding into Software Quality Assurance. As a contractor she has provided software support to major organizations. She provides general Software Quality Assurance direction to the COMATS team which includes generating Test Plans, Test Procedures, Validations Reports and is generally responsible for the overall testing process.

**Address for all three authors is:
Lawrence Livermore National Laboratory
Mail Stop L-347
P.O. Box 808
Livermore, California 94450
FAX: (510) 423-1685**